CS-547Assignment 3

Submitted By:
1801CS66- Abhay Singh,
2111CS04- Hasrat Ali Arzoo
2111CS14- Shashadhar Das
2111MC08- Omkar Gavhane

Q) Prepare a report describing the different Access control methods adopted by MAC (IoS) System.

About Mac-Os:

The Macintosh Operating System (Mac OS) is an operating system (OS) designed by Apple Inc. to be installed and operated on the Apple Macintosh series of computers. Introduced in 1984, it is a graphical user interface (GUI) based OS that has since been released as multiple different versions
Mac OS is considered the pioneer of GUI based operating systems, as it was launched when MS-DOS was the industry standard. Mac OS is a completely capable OS that provides functionality and services similar to Windows or Linux OS. Some of the code base and features of Lisa OS have been incorporated in Mac OS.

Apple MacOS security

For a long time, Mac users didn't have to worry about viruses and malware. Vulnerabilities in the Mac operating system were rarely exploited in the real world. Mac users have always been aware of potential security threats, but much of that was because Windows-using coworkers have been the target of malicious software for ages. The numerous vulnerabilities in every version of Windows in concert with a very large user base made PC users a perfect target.

These days, the potential Mac threat landscape still isn’t as worrisome as on other platforms, but Mac users can no longer afford to ignore the possibility of being compromised by malicious software. These threats will only grow more numerous and more sophisticated as time goes on and more Apple devices are purchased.

It’s happening now: 2017 was a big year for security breaches. In February, a fake Adobe Flash installer carried MacDownloader malware that attempted to transit Keychain data (which includes usernames and passwords, among other personal data). Last autumn, several vulnerabilities were detected in shipping versions of the latest Mac operating system, High Sierra, one granting root access to certain areas without a password prompt. Shortly thereafter, we learned that the processor vulnerabilities called Spectre and Meltdown affect the majority of computers in the world.

Access Control in MAC-OS

- MAC-OS supports security extensions based on the POSIX
- These security mechanisms include file system **Access Control Lists** and **Mandatory Access Control (MAC)**

Access Control List(ACL)

Access Control Lists (ACLs) extend the standard UNIX permission model in a POSIX compatible way.

This permits an administrator to take advantage of a more fine-grained permissions model.

The MAC-OS GENERIC kernel provides ACL support for UFS file systems. Users who prefer to compile a custom kernel must include the following option in their custom kernel configuration file:

```
options UFS_ACL
```

If this option is not compiled in, a warning message will be displayed when attempting to mount a file system with ACL support. ACLs rely on extended attributes which are natively supported in UFS2.

Enabling ACL Support

ACLs are enabled by the mount-time administrative flag,acls, which may be added to /etc/fstab

File systems with ACLs enabled will show a plus (+) sign in their permission settings:

```
drwx------  2 robert  robert  512 Dec 27 11:54 private
drwxrwx---+ 2 robert  robert  512 Dec 23 10:57 directory1
drwxrwx---+ 2 robert  robert  512 Dec 22 10:20 directory2
drwxrwx---+ 2 robert  robert  512 Dec 27 11:57 directory3
drwxr-xr-x  2 robert  robert  512 Nov 10 11:54 public_html
```

In this example,directory1,directory2,and directory3 are all taking advantage of ACLs, whereas private and public_html are not.

Using ACLs

File system ACLs can be viewed using getfacl. For instance, to view the ACL settings on **test**:

```
% getfacl test
        #file:test
        #owner:1001
        #group:1001
        user::rw-
        group::r--
        other::r--
```

To change the ACL settings on this file, use setfacl. To remove all of the currently defined ACLs from a file or file system, include -k

```
% setfacl -k test
```

To modify the default ACL entries, use -m:

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- test
```

Mandatory Access Control(MAC)

Mandatory Access Control (MAC) allows access control modules to be loaded in order to implement security policies

Terminologies associated

level: The increased or decreased setting of a security attribute. As the level increases, its security is considered to elevate as well.

label: A security attribute which can be applied to files, directories, or other items in the system. It could be considered a confidentiality stamp. When a label is placed on a file, it describes the security properties of that file and will only permit access by files, users, and resources with a similar security setting. The meaning and interpretation of label values depends on the policy

configuration. Some policies treat a label as representing the integrity or secrecy of an object while other policies might use labels to hold rules for access.

multilabel: this property is a file system option which can be set in single-user mode using tunefs, during boot using fstab, or during the creation of a new file system. This option permits an administrator to apply different MAC labels on different objects. This option only applies to security policy modules which support labeling.

single label: a policy where the entire file system uses one label to enforce access control over the flow of data. Whenever multilabel is not set, all files will conform to the same label setting.

object: an entity through which information flows under the direction of a *subject*. This includes directories, files, fields, screens, keyboards, memory, magnetic storage, printers or any other data storage or moving device. An object is a data container or a system resource. Access to an object effectively means access to its data

subject: any active entity that causes information to flow between objects such as a user, user process, or system process. On FreeBSD, this is almost always a thread acting in a process on behalf of a user.

More about labels in Mandatory Access Control (MAC)

A MAC label is a security attribute which may be applied to subjects and objects throughout the system

There are two types of label policies: *Single label and Multi label*.

By default, the system will use single label.

A single label security policy only permits one label to be used for every subject or object. Since a single label policy enforces one set of access permissions across the entire system, it provides lower administration overhead, but decreases the flexibility of policies which support labeling. However, in many environments, a single label policy may be all that is required.

A **single label policy** is somewhat similar to **Discretionary Access Control(DAC)** as root configures the policies so that users are placed in the appropriate categories and access levels. A notable difference is that many policy modules can also restrict root. Basic control over objects will then be released to the group, but root may revoke or modify the settings at any time.

When appropriate, a multi label policy can be set on a UFS file system by passing multilabel to tunefs. A multi label policy permits each subject or object to have its own independent MAC label.The decision to use a **multi label** or single label policy is only required for policies which implement the labeling feature, such as **biba,lomac, andmls**.

Some policies, such as **seeotheruids,portacl and partition, do not use labels** at all.

Using a multi label policy on a partition and establishing a multi label security model can increase administrative overhead as everything in that file system has a label. This includes directories, files, and even device nodes command will set multilabel on the specified UFS file system is

```
# tunefs -l enable /
```

Label configuration

All configuration may be done using **setfmac**, which is used to **set MAC labels on system objects**, and **setpmac**, which is used to **set the labels on system subjects**.

For example, to set the biba MAC label to high on test command is

```
# setfmac biba/high test
```

A few MAC-OS policy modules which support the labeling feature offer three predefined labels:**low**,**equal**, and **high**, where:

low is considered the lowest label setting an object or subject may have. Setting this on objects or subjects blocks their access to objects or subjects marked high.

Equal sets the subject or object to be disabled or unaffected and should only be placed on objects considered to be exempt from the policy.

High grants an object or subject the highest setting available in the Biba and MLS policy modules.

Such policy modules include mac_biba,mac_mls and mac_lomac.

The MAC Multi-Level Security Module(MLS)

Module name:**mac_mls.ko**

Kernel configuration line:options MAC_MLS

Boot option:mac_mls_load="YES"

The mac_mls policy controls access between subjects and objects in the system by enforcing a strict information flow policy

Three labels are included in this policy:mls/low,mls/equal, and mls/high, where:

- Anything labeled with mls/low will have a low clearance level and not be permitted to access information of a higher level. This label also prevents objects of a higher clearance level from writing or passing information to a lower level.
- mls/equal should be placed on objects which should be exempt from the policy.
- mls/high is the highest level of clearance possible. Objects assigned this label will hold dominance over all other objects in the system; however, they will not permit the leaking of information to objects of a lower class.

MLS provides:

- A hierarchical security level with a set of non-hierarchical categories.
- Fixed rules of no read up, no write down. This means that a subject can have read access to objects on its own level or below, but not above. Similarly, a subject can have write access to objects on its own level or above, but not beneath.
- Secrecy, or the prevention of inappropriate disclosure of data.

- A basis for the design of systems that concurrently handle data at multiple sensitivity levels without leaking information between secret and confidential.

To manipulate MLS labels, use setfmac. To assign a label to an object:

```
# setfmac mls/5 test
```

To get the MLS label for the file **test**

```
# getfmac test
```

The MAC Biba Module

Module name:**mac_biba.ko**

Kernel configuration line:options MAC_BIBA

Boot option: mac_biba_load="YES"

In Biba environments, an "integrity" label is set on each subject or object

Supported labels are biba/low,biba/equal, and biba/high, where:

- biba/low is considered the lowest integrity an object or subject may have. Setting this on objects or subjects blocks their write access to objects or subjects marked as biba/high, but will not prevent read access.
- biba/equal should only be placed on objects considered to be exempt from the policy.
- biba/high permits writing to objects set at a lower label, but does not permit reading that object. It is recommended that this label be placed on objects that affect the integrity of the entire system.

Biba provides:

- Hierarchical integrity levels with a set of non-hierarchical integrity categories.

- Fixed rules are no write up, no read down, the opposite of MLS. A subject can have write access to objects on its own level or below, but not above. Similarly, a subject can have read access to objects on its own level or above, but not below.
- Integrity by preventing inappropriate modification of data.
- Integrity levels instead of MLS sensitivity levels.

To access the Biba policy setting on system objects, use setfmac and getfmac:

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

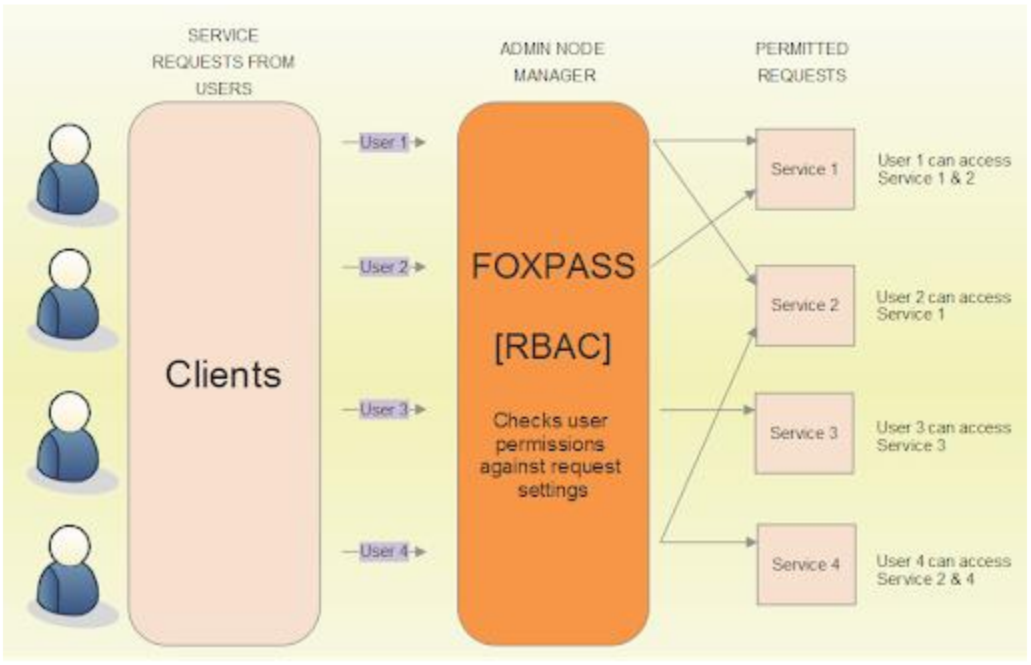
Role-Based Access Control:

Role-based access control (RBAC) is an alternative approach to mandatory access control (MAC) and discretionary access control (DAC) for the purpose of restricting system access to authorized users. RBAC is policy neutral. This makes it more flexible in the provision of access control with many of the features of both Discretionary Access Control (DAC) and Mandatory Access Control (MAC).
RBAC changed the way that authentication is addressed. MAC and DAC were previously regarded as the only models to provide access control. Thus an access model was either a DAC or a MAC model. RBAC is not truly in either category but supports the best features of both.

Role-based access control (RBAC) is a model of access control that, similar to MAC, functions on access controls set by an authority responsible for doing so, rather than by the owner of the resource. The difference between RBAC and MAC is that access control in RBAC is based on the role the individual being granted access is performing. For example, if we have an employee whose only role is to enter data into a particular application, through RBAC we would only allow the employee access to that application, regardless of the sensitivity or lack of sensitivity of any other resource he might potentially access. If we have an employee with a more complex role—customer service for an online retail application, perhaps—the employee's role might require him to have access to information about customers' payment status and information, shipping status, previous orders, and returns, in order to be able to assist said customers. In this case, RBAC would grant him considerably more access. We can see RBAC implemented in many large-scale applications that are oriented around sales or customer service. While this provides much greater granularity of security, it is also much more labor intensive to implement and manage.

Role-based access control grants access privileges based on the work that individual users do. A popular way of implementing "least privilege" policies, RBAC limits access to just the resources users need to do their jobs.
Implementing RBAC requires defining the different roles within the organization and determining whether and to what degree those roles should have access to each resource. Accounts payable administrators and their supervisor, for example, can access the company's payment system. The administrators' role limits them to creating payments without approval authority. Supervisors, on the other hand, can approve payments but may not create them.

The diagram below is a basic overview of RBAC in MAC OS:


SERVICE REQUESTS FROM USERS
ADMIN NODE MANAGER
PERMITTED REQUESTS
Clients
User 1
User 2
User 3
User 4
FOXPASS
[RBAC]
Checks user permissions against request settings
Service 1
Service 2
Service 3
Service 4
User 1 can access Service 1 & 2
User 2 can access Service 1
User 3 can access Service 3
User 4 can access Service 2 & 4

Pros and cons of different Access control methods :

Role Based Access Control:

Pros:

- Flexibility — Administrators can optimize an RBAC system by assigning users to multiple roles, creating hierarchies to account for levels of responsibility, constraining privileges to reflect business rules, and defining relationships between roles.
- Ease of maintenance — With well-defined roles, the day-to-day management is the routine on-boarding, off-boarding, and cross-boarding of users' roles.
- Centralized, non-discretionary policies — Security professionals can set consistent RBAC policies across the organization.
- Lower risk exposure — Under RBAC, users only have access to the resources their roles justify, greatly limiting potential threat vectors.

Cons:

- Complex deployment — The web of responsibilities and relationships in larger enterprises makes defining roles so challenging that it spawned its own subfield: role engineering.
- Balancing security with simplicity — More roles and more granular roles provide greater security, but administering a system where users have dozens of overlapping roles becomes more difficult.
- Layered roles and permissions — Assigning too many roles to users also increases the risk of over-privileging users.

Mandatory Access Control:

Pros:

Enforceability — MAC administrators set organization-wide policies that users cannot override, making enforcement easier.
Compartmentalization — Security labels limit the exposure of each resource to a subset of the user base.

Cons :
Collaboration — MAC achieves security by constraining communication. Highly collaborative organizations may need a less restrictive approach.
Management burden — A dedicated organizational structure must manage the creation and maintenance of security labels.

Access control List:

Pros
The main advantage of ACLs is their simplicity. An ACL clearly lays out the levels of access and permissions that each user, group, or device has on a particular system. This makes it easy to define and interpret an ACL. Since these lists can easily be made human readable, an administrator can easily determine the current permissions and access controls placed on a system, make edits, and revoke permissions as necessary.

Cons :
ACLs lack efficiency since they only support explicitly declared access controls. If, for example, a user has unique access or permissions because they are both in the IT department and a manager, this level of access must be explicitly stated rather than inferred based upon membership in both groups. This requirement for explicit declaration of access controls also impacts scalability. As the number of users, groups, and resources grows, so does the length of the ACL and the time required to determine the level of access granted to a particular user.

Discretionary Access Control:

Pros:

- DAC systems are usually easier to manage

- Very commonly used in UNIX, Windows, Linux, and other network operating systems.

Cons:

- Users can also delete files while reading & writing also.
- DAC systems can be a little less secure than MAC systems

macOS: The security mechanisms in macOS include file system **Access Control Lists** and **Mandatory Access Control (MAC).** whose pros and cons are given above

Windows NT: Windows NT operating system provides a standard representation of access control lists (ACLs) for use.
Pros and Cons of file Access control list has already been mentioned above.

Linux: Linux uses Mandatory access control ,Access control list and DAC pros and cons of which are mentioned above.

Comparison of Access Control method of some Operating Systems:

Linux vs Windows NT

Linux: The Linux file system derives its implementation from Unix, which was released in 1973 and is much older than Windows. When Unix was first designed by AT&T, disk space was at a premium, and each bit on the hard drive mattered. In order to maximize disk space, each file could only have three sets of permissions – access permissions for everyone, access permissions for its “owning group”, and access permission for its “owning user”. Each of those three permissions could be defined as either Read, Write, or Execute.

Windows NT: Windows NT supports multiple file systems, but the protection issues we will consider are only associated with one: NTFS. In NT there is the notion of an item, which can be a file or a directory. Each item has an owner. An owner is usually the thing that created the item. It can change the access control list, allow other accounts to change the access control list and allow other accounts to become owner. Entries in the ACL are individuals and groups. Note that NT was designed for groups of machines on a network, thus, a distinction is made between local groups (defined on a particular workstation) and global groups (domain wide). A single name can therefore mean multiple things.

NTFS is structured so that a file is a set of properties, the contents of the file being just one of those properties. An ACL is a property of an item. The ACL itself is a list of entries: (user or group, permissions). NTFS permissions are closer to extended permissions in UNIX than to the 9 mode bits.

macOS vs Windows 10:

Windows 10: Windows has many features that provide integrity to the OS and user data files. Microsoft Windows Millennial Edition (Windows ME) introduced an OS file protection process called System File Protection (SFP). If anything deleted a system critical file, SFP ensured that Windows would immediately replace it with a known good copy. Windows Vista introduced a version of SFP known as Windows Resource Protection, which also protected critical Windows registry settings, although what was protected and automatically replaced diminished overall.

Vista also introduced Mandatory Integrity Controls (MIC) and file and registry virtualization. With MIC, every user, file, and process in Windows is explicitly assigned a MIC level (high, medium, low). Users, files, and processes of lower MICs cannot modify objects of higher MICs.

macOS: Introduced in El Capitan in 2015, the security feature called System Integrity Protection (SIP) addresses the problem with unrestricted root access if malware or hackers gain access to the account credentials. SIP protects the contents and permissions of certain important files and directories, even from actions performed as root. SIP protects against running unsigned kernel extensions, and it protects processes against code injections and real-time modifications to code without specific entitlements. Only properly signed apps can modify the protected system directories, and those apps must be tied to a developer ID and with entitlements signed by Apple.

References:

- https://www.cse.wustl.edu/~jain/cse571-14/ftp/ios_security/index.html
- https://ieeexplore.ieee.org/document/9152695
- https://westoahu.hawaii.edu/cyber/best-practices/best-practices-weekly-summaries/access-control/#:~:text=Three%20main%20 types%20of%20 access,Mandatory%20Access%20Control%20(MAC).&text=DAC%20is%20a%20type%20of,on%20rules%20specified%20by%20users.
- https://is.muni.cz/th/uny2u/xcsanyi_bc.pdf
- http://www.cs.cornell.edu/courses/cs513/2000SP/L07.html